09/091735

Word or phrase: cryptographic key [Search]

[✓ Find definitions] Find translations Search all dictionaries

Descriptions: Verbose, **Compact**
Jump to: General, Art, Business, Computing, Medicine, Miscellaneous, Religion, Science, Slang, Sports, Tech, Phrases

We found 2 dictionaries with English definitions that include the word *cryptographic key*:
Tip: Click on the first link on a line below to go directly to a page where "cryptographic key" is defined.

General (1 matching dictionary)

1. Cryptographic key : Wikipedia, the Free Encyclopedia [home, info]

Computing (1 matching dictionary)

2. cryptographic key : Glossary of Communications, Computer, Data, and Information Security Terms [home, info]

Encyclopedia article (***Cryptographic key***)

A **cryptographic key** is a relatively small amount of information that is used by a cryptographic algorithm to 'customize' the transformation of plaintext into cyphertext (during encryption) or from cyphertext to plaintext (during decryption). (continued)

Introducing OneLook's unique Word of the Day feature.

Additional links for *cryptographic key*...

Search completed in 0.144 seconds.

Cryptographic key

From Wikipedia, the free encyclopedia.

A **cryptographic key** is a relatively small amount of information that is used by a cryptographic algorithm to 'customize' the transformation of plaintext into cyphertext (during encryption) or from cyphertext to plaintext (during decryption). The same algorithm and plaintext, but with a different key will produce a quite different cyphertext, and so for decryption as well. If the decryption key is lost, encrypted data will not in practice be convertible back to its original form -- at least for high quality encryption algorithms and large enough key sizes. Thus, the security of a cryptographic key in most cases relies on its being kept secret: hence the alternative name **secret key**.

Most cryptographic algorithms use a single key for both encryption and decryption: they are known as symmetric key algorithms. An attacker who obtains the key (by theft, extortion, dumpster diving, or inspection of a Post-It note stuck to the side of a terminal) can recover the original message from the encrypted data, since as a matter of principle the details of the cryptographic algorithm used is assumed to be already available to the attacker. This design assumption is usually known to cryptanalysts as Kerckhoffs' law, or in more colloquial form, Shannon's Maxim. It is fully justified by long and painful practical experience over some thousands of years and no recent development has changed this reality. That secrecy of a crypto system is important (or even vital) is widely, and wrongly, believed. As a general principle one would not want one's crypto system to be fully known to the opposition, but it should remain secure even if the opposition learns all about it. The chances are excellent that they will anyway.

A new class of cryptographic encryption algorithms was discovered in the 1970s which use a pair of keys, one to encrypt and one to decrypt. Some of these asymmetric key algorithms have the property that it is not possible to determine one key from the other (so far as is currently known). Such an algorithm allows one key to be made public while retaining the private key in only one location.

Key Sizes

Typical key sizes for estimated 'equivalent security' against a particular kind of attack (ie, brute force key space search) are 128 bits for symmetric ciphers and 2048 bits or more for public key cryptography. Elliptic curve cryptography may allow much smaller size keys for equivalent security, but these algorithms have only been known for a relatively short time and current estimates of the difficulty of brute force searching for their keys may not survive. Recently, a message encrypted using a 109-bit key elliptic curve algorithm was broken by brute force. As a result it would appear that elliptic curve algorithm keys must be somewhat the same length as symmetric key algorithm keys for equivalent security. As always, for all but the one-time pad, a theoretical breakthrough may make everything you've encrypted an open book regardless of the algorithm or algorithm type you've chosen, and a too-short key will certainly do so.

If the key is too small, the algorithm will be vulnerable to a brute force attack in which all possible values of the key are tried one by one. 'Birthday' attacks are also possible; the probability of a 'collision' between a large group of values goes up roughly as the square of the number of possible values and this applies in cryptography as well. In addition, many algorithms permit reduced effort attacks as compared to brute force key search. If the effort is sufficiently reduced, the algorithm will be 'insecure' against that attack and should not be used. It may be expected that algorithms for which no improved attack is now known, and for which a brute force attack is impractical, will be found to be insecure when some new cryptoanalytic technique is developed. When one is.

The problem of choosing a cryptographic algorithm reduces itself, in actual practice, to an estimate of how likely such an advance will be over the relevant time. Personal secrets need to be kept confidential for different durations than tactical deployment information in a battle, and still differently than some commercially valuable information (eg, the formula for Coke). There are no good answers known to this problem. Intelligent, cryptographically informed, choosers limit their choice to publicly known and publicly unbroken, but well studied, algorithms. Only algorithms from this group can be credibly thought secure. All others are either not sufficiently well tested, or are from secret organizations with adequate testing resources, but also with ulterior motives.

Key Choice

At the least sensible, choosing a key by increasing the value of the last used key by one is clearly foolish. Any attacker noticing the key choice pattern will be ecstatic. In fact, experience has shown that pattern in key choice are a very very significant source of breaks into otherwise well designed crypto systems. The Japanese Purple cypher machine of WWII is an example, for after the initial breakthrough by US cryptanalysts, the poor choice of keys made continuing breaks into the Purple traffic very much easier.

In general, keys _must_ be chosen randomly (or alternatively, they must be random values) while meeting other requirements of the algorithm in use. This is a fundamentally difficult, quite subtle, problem and has been 'solved' in one or another crypto system in various ways. There is an Internet RFC on generating randomness (RFC 1750, Randomness Recommendations for Security), but it is long on prescription and short on explanation. In general randomness is always a problem in cryptography, and key choice is merely another example.

Failure to handle this properly is an easy way to render any cryptosystem insecure. See randomness.

Applications

Pretty Good Privacy (PGP) is a popular program that intelligently uses both symmetric and asymmetric algorithms as part of an excellent crypto system design. PGP uses the timing between keystrokes to generate 'randomness'; thus far this has not been found unsatisfactory. A public Standard has been recently adopted for a PGP compatible crypto system. OpenPGP is the standard and GPG is an implementation of it available from the Free Software Foundation. There is a Web site for the FSF which has pointers to the official Web pages for both PGP and OpenPGP.

External links

- http://www.pobratyn.com/pk.php3 An example of a PGP Public Key. -- broken link
- The official site for PGP International and, unofficially, for GPG.
- Crypto-Gram, a monthly online publication on cryptography and its sensible use.

Edit this page | Discuss this page | Page history | What links here | Related changes
Other languages: Deutsch | Svenska
Main Page | About Wikipedia | Recent changes | Go
Search

WIKIPEDIA
The Free Encyclopedia

Symmetric key algorithm

From Wikipedia, the free encyclopedia.

A **symmetric-key algorithm** is an algorithm for cryptography that uses the same cryptographic key to encrypt and decrypt the message. (Actually, it is sufficient for it to be easy to compute the decryption key from the encryption key and vice versa.) Other terms for symmetric-key encryption are *single-key* and *private-key* encryption. Use of the latter term is discouraged because of conflict with the term *private key* in public key cryptography.

Symmetric-key algorithms can be divided into stream ciphers and block ciphers. Stream ciphers encrypt the bits of the message one at a time, and block ciphers take a number of bits and encrypt them as a a single unit. Blocks of 64 bits have been commonly used; the Advanced Encryption Standard algorithm approved by NIST in December 2001 uses 128-bit blocks.

Symmetric-key algorithms are generally much faster to execute electronically than asymmetric key algorithms. The disadvantage of symmetric-key algorithms is the requirement of a *shared secret key*.

Asymmetric key algorithm

From Wikipedia, the free encyclopedia.

In cryptography, an **asymmetric algorithm** uses a pair of different cryptographic keys to encrypt and decrypt the plain text. Typically, the two keys are related mathematically; a message encrypted by the algorithm using one key can be decrypted by the same algorithm using the other. In a sense, one key "locks" a lock (encrypts); but a different key is required to unlock it (decrypt).

One analogy which can be used to understand the advantages of an asymmetric system is to imagine two people, Alice and Bob, sending a secret message through the public mail.

In a symmetric key system, Alice first puts the secret message in a box, and then padlocks the box using a lock to which she has a key. She then sends the box to Bob through regular mail. When Bob receives the box, he uses an identical copy of Alice's key (which he has obtained previously) to open the box, and reads the message.

In an asymmetric system, instead of opening the box when he receives it, he simply adds his own personal lock to the box, and returns the box through public mail to Alice. Alice uses her key to remove her lock, and returns the box to Bob, with Bob's lock still in place. Finally, Bob then uses his key to remove his lock, and reads the message from Alice.

The critical advantage in an asymmetric system is that Alice never needs to send a copy of her key to Bob. This reduces the possibility that a third party (for example, an unscrupulous postmaster) will copy the key while it is in transit to Bob, allowing that third party to spy on all future messages sent by Alice. In addition, if Bob is careless and allows someone else to copy *his* key, Alice's messages to Bob will be compromised, but Alice's messages to other people will remain secret.

Not all asymmetric algorithms operate in precisely this fashion. The most common have the property that Alice and Bob each own *two* keys; and one key is not (so far as is known) deducible from the other. These are the 'public key / private key' algorithms, since one key of the pair can be published without affecting security of messages. In our analogy above, Bob might publish instructions on how to make a lock ("public key"); but the lock is of a type where it is very difficult to deduce from these instructions how to make a key which will open that lock ("private key"). Those wishing to send messages to Bob use the public key to encrypt the message; Bob uses his private key to decrypt it.

Of course, there is the possibility that someone could "pick" either of Bob's or Alice's locks. Unlike the case with the one-time pad or its equivalents, there is no currently known asymmetric key algorithm which has been *proven* to be secure against a mathematical attack. That is, it is not known to be impossible that some relation between the keys in a key pair, or a weakness in an algorithm's operation, might be found which would allow decryption without either key, or using only the encryption key. Their security is based on estimates of how difficult the underlying mathematical problem is to solve; and such estimates have changed with both increasing computer power, and new mathematical discoveries.

Weaknesses have been found for promising asymmetric key algorithms in the past. The 'knapsack packing' algorithm was found to be insecure when an unsuspected attack came to light. Recently, some attacks based on careful measurements of the exact amount of time it takes a known piece of hardware to encrypt plain text have been used to simplify the search for likely decryption keys. Thus, use of asymmetric key algorithms does not ensure security; and it is an area of active research to discover and protect against new and unexpected attacks.

The first known asymmetric key algorithm was invented by Clifford Cocks of GCHQ in the UK. It was not made public at the time, and it was reinvented by Rivest, Shamir and Adelman at MIT in the 1970s. It is usually referred to as RSA as a result. RSA relies for its security on the difficulty of factoring very large integers. A breakthrough in that field would call into question RSA's security. Currently, RSA is vulnerable to an attack by factoring the modulus part of the public key, even when keys are properly chosen, for keys shorter than perhaps 700 bits. Most authorities suggest that 1024 bit keys will be secure for some time, barring a fundamental breakthrough in factoring practice, but others favor even longer keys.

At least two other algorithms were invented after the GCHQ work, but before the RSA publication. These were the Ralph Merkle puzzle cryptographic system and the Diffie-Hellman system.

A relatively new addition to the class of asymmetric key algorithms is elliptic curve cryptography. While it is more complex computationally, it is believed by many to represent a more difficult mathematical problem than either the factorisation or discrete logarithm problems which form the basis of other popular asymmetric algorithms.

One drawback of asymmetric key algorithms is that they are much slower (factors of 100+ are typical) than comparably secure symmetric key algorithms. In many quality crypto systems, both algorithm types are used. The receiver's public key encrypts a symmetric algorithm key which is used to encrypt the main message. This combines the virtues of both algorithm types when properly done.

Public-key cryptography

(Redirected from Public key cryptography)

Asymmetric-key cryptography, also known as **public-key cryptography**, is a form of cryptography in which asymmetric key algorithms are used for encryption, signature, etc. In these algorithms, one key is used to encrypt a message and another is used to decrypt it, or one key is used to sign a message and another is used to verify the signature. The key used to decrypt or sign must be kept secret ('private') and cannot (so algorithm designers hope) be derived from the public key, which is used to encrypt or verify, and which may be known to any.

Several asymmetric key algorithms have been developed beginning in the 1970s. One widely-used algorithm is RSA. It uses exponentiation modulo a product of two large primes to encrypt and decrypt. The public key exponent differs from the private key exponent, and determining one exponent from the other is fundamentally hard without knowing the primes. Another is ElGamal (developed by Taher ElGamal) which relies on the discrete logarithm problem. A third is a group of algorithms based on elliptic curves, first discovered by Neil Koblentz in the late 80.

Note that there is *nothing* special about asymmetric key algorithms. There are good ones, bad ones, insecure ones, etc. None have been proved 'secure' in the sense the one-time pad has, and some are known to be insecure (ie, easily broken). Some have the public key / private key property in which one of the keys is not deduceable from the other; or so it is believed by knowledgeable observers. Some do not, it having been demonstrated that knowledge of one key gives an attacker the other. As with all cryptographic algorithms, these must be chosen and used with care.

Public-key algorithms can be used for either confidentiality or sender authentication. A user can encrypt a message with their private key and send it. That it can be decrypted by the public key provides assurance that that user (and no other) sent it. Unless the private key has been compromised, of course. These algorithms can also be used to for confidentiality, a message which is encrypted by the receipient's public key can only be decrypted by a person in possession of the paired private key.

Examples of well regarded asymmetric key algorithms include:

- RSA encryption algorithm
- ElGamal
- elliptic curve encryption algorithms

examples of not well regarded asymmetric key algorithms include:

- Merkle-Hellman the 'knapsack' algorithms

examples of protocols using asymmetric key algorithms include:

- DSS Digital Signature Standard which incoporates the Digital Signature Algorithm
- PGP
- GPG an implementation of OpenPGP
- ssh
- Secure Socket Layer now implemented as an IETF standard -- TLS

See also: GNU Privacy Guard, Pretty Good Privacy, Secure Sockets Layer, Secure Shell, pseudonymity, Quantum cryptography, Key escrow, public key infrastructure (PKI).